基于多进程并行加速的太阳高分辨图像重建方法*

邓涛[1]， 陈东[2]， 代红兵[1]，王新华[2,3]
(1.云南大学 信息学院 云南 昆明 650504;
2.中国科学院云南天文台 云南 昆明 650011;
3.中国科学院大学 北京 100049)

摘 要：目前太阳高分辨图像重建往往采用斑点干涉术和斑点掩模法重建目标的模和相位，由于分组分块数据量大，算法复杂等因素，难以满足实时重建的要求。为了能缓解数据处理的压力，在现有在单组分块数据CPU/GPU混合计算方法的基础上，提出通过多进程将多组分块数据分配到GPU上同时并行处理的方法。实验表明，基于多进程并行加速方法可提高CPU和GPU资源利用效率，使GPU能同时处理多组分块数据，可显著提升图像分块处理的速度，加速比达4.7左右，相关研究可为天文数据并行化处理提供借鉴参考。

关键词：多进程；并行计算；图像重建；斑点掩模法；斑点干涉法

中图分类号： P182.2+1 TP751.2 **文献标识码：**A

1 引 言

太阳是影响人类活动的最大一颗恒星，尤其是太阳活动对地球环境、气候和天气的影响[1]等。由于太阳光线在穿过地球大气层时受到大气湍流的影响会产生波前误差，使得光路发生偏转，观测的太阳图像出现不同程度的偏移、抖动、模糊等形态，导致地基式望远镜的观察图像质量下降[2]。为了消除大气湍流对望远镜成像结果的影响，通常采用空间望远镜、自适应光学以及图像重建技术等方式获取太阳高分辨图像。

目前图像重建技术常用的算法主要有相位差法、多帧盲反卷积法 、斑点干涉术、K-T 算法、斑点掩模法、简单位移叠加法 、迭代位移叠加法以及选帧位移叠加法等。这些算法都是通过大量的短曝光图像重建出太阳高分辨图像，但往往因为处理的数据量很大 、算法复杂导致无法达到实时重建的需求。

近年，国内外在太阳高分辨图像重建算法并行化研究方面做了很多工作，并获得一定的加速比。Denker[3]等人在 BBSO 基于双节点集群，提出一种帧选择和斑点掩模法的并行计算方法，采用多线程方法在 57s 内完成重建一张 256×256pixel 图像。Friedrich Wöger[4]等人使用 ATST，利用改写于 KISIP 的算法，采用 GPU 一次处理单组分块数据，实现 4.2s 内重建 225 个 128×128 pixel 子块的相位。Li Xuebao[5]等人从 NVST 的 TiO 通道中选取一组子块图像（100×256×256 pixel）。采用 OpenMP 方法实现了一组子块图像的并行化，重建单帧 256×256 pixel 的子块图像，运行时间减少至 2.7s，获得 2.5 倍加速. 郑艳芳[6]等人采用 GPU 的 CUDA 构架对光球 TiO 通道中的一组子块（100×256×256 pixel）实现并行化，基于 CUDA 方法重建单组子图像组的运行时间减少到大约 0.7s。宣经纬[7]等人基于 CUDA 并行计算架构，在斑点掩模算法中实现单个子块 GPU 内的并行化，采用 CUDA 并行方法比纯 CPU 上运行的串行算法加速比达 7。

*基金项目：国家自然科学基金(U1831210)资助.
作者简介：邓涛，男，硕士研究生。研究方向：计算机技术。Email: 871604689@qq.com
通讯作者：陈东，男，高级工程师，博士。研究方向：天文仪器及方法。Email: chd@ynao.ac.cn

综上所述，太阳高分辨图像重建在采用斑点掩模法下的图像并行化研究方面已经取得一定的研究成果，但大多还局限于单纯的 CPU 或 GPU 加速，而且 GPU 一次只能处理一组分块数据，CPU/GPU 的并行化能力没有完全发挥出来。如何将多组分块数据分配到 GPU 上同时并行处理，进一步提高 CPU 和 GPU 的并行计算能力和资源效率，本文提出了基于多进程并行加速太阳高分辨图像重建方法。

2 太阳高分辨图像重建方法

一米新真空太阳望远镜（New Vacuum Solar Telescope, NVST）[8]坐落在抚仙湖畔，主要的观察波段有：G-band (4305 Å)、Hα (6562.8 Å)、TiO (7058 Å)。NVST 太阳高分辨图像重建使用了两个级别：Level1 级别是采用位移叠加法来重建色球图像，Level1+则是采用斑点掩模法重建光球或色球图像。Level1+级别计算复杂度比 Level1 高，但 Level1+级别在视宁度好得时候重建的图像质量更好、信噪比更高。

NVST Level1+级别太阳高分辨图像重建流程：图像预处理，图像初对齐，视宁度估计，图像分块处理，子块拼接，其中图像分块处理主要采用斑点干涉术和斑点掩模法重建太阳高分辨图像的模和相位。图像分块处理包含若干处理环节，其中相位递推环节由于数据量大、计算复杂，并且需要考虑多个递推路径的整合，使得图像分块处理在 Level1+级别的重建流程中最为耗时，因此子块处理的好坏严重影响重建的效果和时效，其中最关键过程描述如下：

2.1 振幅重建

在满足等晕区情况下，短曝光像是目标和系统的点扩展函数的卷积：

$$\mathrm{i}\left(x,y\right)=o(x,y)\otimes p(x,y) \tag{1}$$

上式$\otimes$为卷积符号，时域中的卷积对应于频域中的乘积。上式在频域中满足：

$$I\left(u,v\right)=O(u,v)*P(u,v) \tag{2}$$

其中$I\left(u,v\right)$，$O(u,v)$，$P(u,v)$分别是时域中对应项的傅里叶变换

功率谱统计为：

$$\left\langle\left|I\left(u,v\right)\right|^2\right\rangle=\left\langle\left|O(u,v)\right|^2\right\rangle*\left\langle\left|P(u,v)\right|^2\right\rangle \tag{3}$$

其中$\langle\ \rangle$为系综平均，$\left\langle\left|P(u,v)\right|^2\right\rangle$就是 SITF. 其中 SITF 可以有两种方法得到：1. 观测目标的近参考星多幅斑点图的平均功率谱建模计算得到 SITF。2. 使用谱比法计算大气相干长度r_0，从而使用功率谱退卷积得到 SITF。

2.2 相位重建

目标斑点图的三重相关为：

$$i^{(3)}(x,x')=\int i(x'')i(x'',x')i(x'',x')dx'' \tag{4}$$

目标斑点图的三重自相关的傅里叶变换有：

$$I^{(3)}(u,v)=I(u)I(v)I(-u,-v) \tag{5}$$

其中$I(u)$是$i(x)$的傅里叶变换。u和v是二维空间频率。

目标斑点图的平均重谱有：

$$\left\langle I^{(3)}(u,v)\right\rangle = O^{(3)}(u,v)\left\langle P^{(3)}(u,v)\right\rangle \tag{6}$$

其中，<>为取其系综平均。$\left\langle P^{(3)}(u,v)\right\rangle$ 为平均斑点掩模法传递函数。

在得到平均重谱后，由低频到高频的相位元逐步递推恢复其目标斑点图全部相位。

3 多进程并行加速方法

3.1 CPU/GPU 混合计算方法

为了满足整个视场线性空间平移不变性，需要把预处理和初对齐后的图像分割成一个个子块，然后把相同位置上的子块合并成子块组。NVST Level11+级别的太阳高分辨图像重建中现有并行加速方法大多还局限于 CPU/GPU 混合计算方法，即 GPU 一次只能处理单个子块组，不同子块组之间依然还是串行计算。基于 CPU/GPU 混合计算方法的图像分块处理流程如图 1 所示。

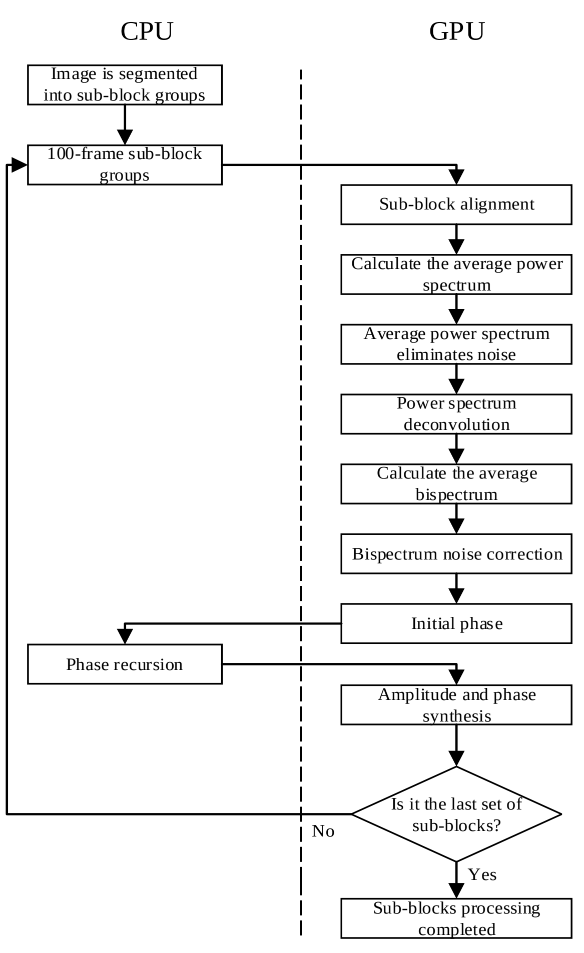


图 1 基于 CPU/GPU 混合计算方法的图像分块处理流程

Figure 1 Image block processing flow based on CPU/GPU hybrid computing method

按照等晕区分块后，每一个子块组都含有多帧子块，在处理一个子块组时，除了相位递推计算比较复杂适合在 CPU 端完成外，其他都是在 GPU 端完成。相位递推需要用到四维重谱[9]，

从低频相位点到高频相位点依次遍历全部有效的相位点。每个相位点都是由多个相位点和对应的重谱计算得到，计算复杂度高。在遍历的过程中，相位递推伴随着大量的逻辑判断，导致适合在 CPU 端处理。

虽然以上方法比纯 CPU 端在图像分块串行处理的处理时间少，但这种方法存在两个问题：一是子块组间的处理是依次进行的（串行），在 GPU 处理时，CPU 存在空闲状态，CPU 利用率不高；二是 GPU 上一次只处理一个子块组，GPU 利用率不高。为此，本文提出了基于多进程对图像分块处理并行加速方法。

3.2 多进程并行加速方法

针对以上现有方法存在的问题，引入多进程，提升 CPU 利用率，并且通过多进程让 GPU 处理更多的子块组，同时提高 CPU 和 GPU 的并行化处理能力。将待处理多帧图像分割成很多子块后，子块合并为若干子块组，把所有的子块组加入任务列表中。每个进程都顺序选择任务列表中的一个子块组，多个进程同时操作 GPU 并行处理多组子块组。基于多进程并行加速方法的图像分块处理流程如图 2 所示。

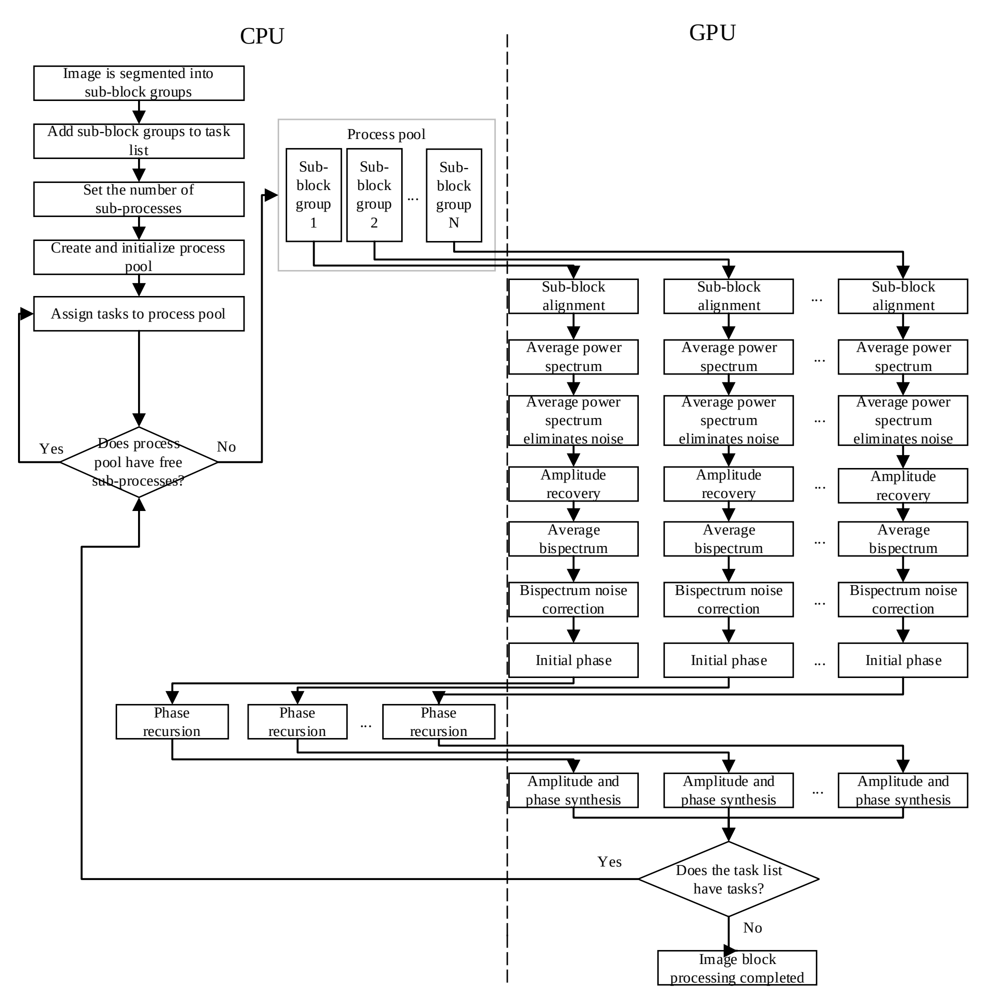


图 2 基于多进程并行加速方法的图像分块处理流程

Figure 2 Image block processing flow based on multi-process parallel acceleration method

多进程并行加速方法的图像分块处理流程如下：

1、创建任务列表。创建并初始化列表，将图像分块后的子块组加入到任务列表中。

2、创建进程池。创建并初始化一个进程池，并在进程池中可以添加适当的子进程数量。

3、分配进程任务。由主进程依次把任务列表中的任务分配给处于空闲状态下的子进程。

4、传递参数。将处于 CPU 端的任务数据传递到 GPU 端，以及子块组处理过程需要的其他参数。

5、处理子块组。进程池开启多少子进程，就有多少个子块组同时并行计算。子块对齐、模的重建、平均重谱的计算以及初始相位的计算等过程都是在 GPU 端并行完成。GPU 端完成初始相位计算后，把初始相位数据传递给 CPU 端的相应子进程开始相位递推，完成相位递推后，子进程将相位数据从 CPU 端传递到 GPU 端。最后进行模和相位的合成。

6、再次分配任务。每个处理完子块组后处于空闲状态的子进程，并不会被销毁而是返回进程池，进程池将空闲子进程信息反馈给主进程，若子块组任务列表仍有待执行的子块组，主进程将待执行的子块组分配给进程池中的空闲子进程。

7、关闭进程池。主进程反复检测子块组任务列表，当任务列表中没有待处理子块组时，并等待所有子进程执行完毕，进程池将所有空闲子进程信息反馈给主进程，释放所有子进程并关闭进程池。

由上述过程可知，各个子块组处理相互独立、互不影响，进程池中每个子进程处理相应的子块组同时并行运行。子块对齐、模的重建、初始相位计算以及模和相位的合成都是由多个子进程在 GPU 端并行完成的，但每个子块组的相位递推是由相应的子进程在 CPU 端并行计算的。值得注意的时，在创建和初始化进程池时，进程并不是越多越好，创建新进程会耗费系统资源，所以进程池中只能添加合适数量的子进程，其数量取决于 CPU 和 GPU 的资源能力。此外，为避免不必要的系统开销，并不立即撤销完成任务的子进程，而是在任务列表还有任务时选择进程复用，再次分配任务。

下面以 Python 为例，其实现方法如下。多进程管理 Multiprocessing 模块提供了 Pool 类，通过 import 命令导入 Multiprocessing 模块，使用 multiprocessing.Pool 函数导入 Pool 类。创建并初始化任务列表 blocklist =[]，将子块组加入到任务列表中。设置指定的工作子进程数目(num)，使用 pool=Pool(num)，按照指定数量子进程创建和初始化进程池，供用户调用。使用 pool.apply_async()、pool.map()、pool.apply()、pool.map_async()等方法，调用子块组处理函数并传递子块组任务列表参数，然后提交给进程池。当新子块组任务请求提交到进程池时，如果 Pool 不满，则会创建一个新进程来执行该任务请求，如果 Pool 满了，告知请求等待。通过 Cupy 模块中 cupy.asarray()方法和 cp.asnumpy()方法来进行 CPU 端和 GPU 端数据传输。每个子块组计算完成使用列表收集结果。当任务列表没有待处理子块组时，使用 pool.close()方法，使进程池不再接受新的任务，当所有子块组计算完成后，工作进程会退出。最后使用 pool.join()方法，等待进程池中所有的子进程结束完毕结束子进程，返回主进程。关键过程的代码实现如下表 1 所示。

表 1 关键过程的代码实现

Table 1 Code implementation of key processes

Code implementation of key processes

```
from multiprocessing import Pool
import cupy as cp
def block_process(cubesub):
      cuberc_gpu=cp.asarray(cubesub)            # Transfer subblock groups from the CPU to the GPU
      # Sub-block alignment, Reconstruction of amplitude s, Calculation of bispectrum, Initial phase
      phalxp_cpu =cp.asnumpy(x)                 # Phase recursion transfers data X from the GPU to the CPU
                                                # phase recursion
      phatmp_gpu=cp.asarray(y)
                                                # Amplitude and phase synthesis
if __name__ == '__main__':
      blocklist =[]                             # Create and initialize task lists
      blocklist.append(cubesub)                 # Add subblock group tasks to the task list
      num=n                                     # Set the number of n sub-processes
      pool = Pool(num)                          # Create and initialize process pools
      for i in blocklist:
            pool.apply_async(block_process,(i,))  # Commit functions and arguments to the process pool
      pool.close()                              # Close the process pool and accept no more tasks
      pool.join()                               # Block waits for all tasks to complete before continuing
```

4 结果和分析

4. 1 实验环境

硬件：Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz(2801 MHz)的处理器（6 核），8GB RAM，NVIDIA GeForce GTX 1050 Ti，4095 MiB。软件：Microsoft Windows 10，Spyder 4.0.13，CUDA 10.2，Python 3.7.6。

4. 2 并行加速结果

以图像分块处理为例，对第三章两种方法进行测时比较，实验结果如下表 2 所示。

表 2 两种方法处理时间

Table 2 Processing time of two methods

Parallelization method	Ha-band image block processing average time (s)	TiO-band image block processing average time (s)
CPU/GPU hybrid computing method	740.68	957.31
Multi-process parallel acceleration method	157.76	203.32

为了验证代码的正确性，实验数据选取 2020 年 6 月 6 日 NVST Ha 波段的观测数据 10 组，每组选取 100 帧（1028×1024 pixel），每帧采用重叠方式分割成 96×96 pixel，将每帧相同位置对应的子块合并成一组，划分为 625 组子块组。实验数据还选取 2020 年 6 月 8 日 NVST TiO 波段的观测数据 10 组，由于受到显存大小的限制，每组选取 50 帧（2160×2560 pixel），每帧采用重叠方式分割成 128×128 pixel。将每帧相同位置对应的子块合并成一组，划分为 1050 组子块组。

实验表明，Ha 波段和 TiO 波段数据在采用多进程并行加速方法时图像分块处理的平均处理时间相对较少，采用多进程并行加速方法在 Ha 波段和 TiO 波段数据的图像分块处理部分中加速比分别可达 4.69 和 4.71。通过大量实验结果分析，由于在 CPU/GPU 混合计算方法中，子块组之间仍然是串行计算，使得 GPU 一次只能处理单组分块数据，而在多进程并行加速方法中，子块组之间是并行计算，使得 GPU 能同时处理多组分块数据。由此可以说明基于多进程并行加速方法可提高 CPU 和 GPU 资源利用效率，可显著提升图像分块处理的速度。

在多进程并行加速方法中，不同的进程数有不同的图像分块处理时间如图 3 所示。

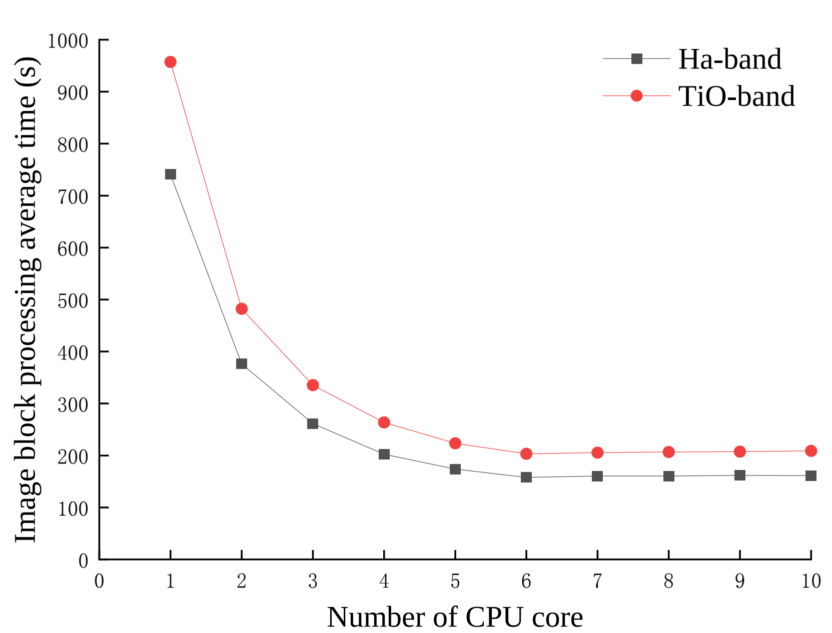


图 3 图像分块处理在不同进程数下的耗时情况

Figure 3 The time consuming of image block processing under different number of processes

图 3 显示不同的进程数对图像分块处理并行加速的耗时情况，随着进程池中子进程数量的增加，Ha 波段的所有子块处理的时间降低到 157.76s，TiO 波段的所有子块处理的时间降低到 203.32s，并行加速的效果明显。但是在使用 6 个进程以后，受到 CPU 核心数、GPU 显存大小以及多进程调度开销的影响，图像分块处理时间会上下浮动并呈现上升的趋势。在选择合适的子进程后，使用多进程并行加速方法处理时，CPU 和 GPU 利用率都提高了，CPU 利用率可以达到 100%。

5 结语

针对现有方法子块组间低效的串行处理而导致 CPU 和 GPU 利用率都不高的问题，本文提出了基于多进程并行加速太阳高分辨图像重建的方法，利用多核和多进程技术，有效提升了 CPU/GPU 的利用率和重建过程速度，其研究可为天文数据并行化处理提供借鉴参考。要进一步提高 CPU/GPU 的并行化程度，仍然有许多亟待突破的关键问题，其中 CPU 端承担的相位递推压力最大，耗费图像分块处理过程 80%的时间，下一步将考虑基于相位递推的特点进行并行化，同时对相关算法进行优化改进，使 CPU/GPU 计算负载达到均衡，是下一阶段研究的重点。此外，相关研究还需要在 MPI/GPU 异构环境中进行验证。

参考文献

[1] 丁一汇. 太阳活动对地球气候和天气的影响[J]. 气象, 2019, 45(3): 297-304. Ding Yihui. Effect of solar activity on earth's climate and weather [J]. Meteorological Monthly, 2019, 45(3): 297-304.
[2] Van Noort M, Van Der Voort L R, Löfdahl M G. Solar image restoration by use of multi-frame blind de-convolution with multiple objects and phase diversity[J]. Solar Physics, 2005, 228(1-2): 191-215.
[3] Denker C, Yang G, Wang H. Near real-time image reconstruction[J]. Solar Physics, 2001, 202(1): 63-70.

[4] Wöger F, Ferayorni A, Radziwill N M, et al. Accelerated speckle imaging with the ATST visible broadband imager[J]. Proceedings of SPIE - The International Society for Optical Engineering, 2012, 8451:84511C-84511C-9.
[5] Li X B, Zheng Y F. A novel parallel method for speckle masking reconstruction using the OpenMP[J]. Journal of The Korean Astronomical Society, 2016, 49(4): 157-162.
[6] Zheng Y F, Li X B, Tian H F, et al. GPU-accelerated speckle masking reconstruction algorithm for high-resolution solar images[J]. Journal of The Korean Astronomical Society, 2018, 51(3): 65-71.
[7] 宣经纬,饶长辉,钟立波,等.基于 GPU 的太阳图像斑点重建技术实现[J].大气与环境光学学报,2020,15(02):90-100. Xuan Jingwei, Rao Changhui, Zhong Libo, et al. Implementation of solar speckle image reconstruction based on GPU[J]. Journal of Atmospheric and Environmental Optics, 2020, 15(02): 90- 100.
[8] Liu Z, Xu J, Gu B Z, et al. New vacuum solar telescope and observations with high resolution[J]. Research in Astronomy and Astrophysics, 2014, 14(6): 705
[9] 向永源. 太阳高分辨高速重建算法的研究[D]. 昆明：中国科学院云南天文台，2016. Xiang Yongyuan. Research on high speed high resolution solar image reconstruction algorithm[D]. Kunming: Yunnan Observatory, Chinese Academy of Sciences, 2016.

High-resolution solar image reconstruction method based on multi-process parallel acceleration

Deng Tao[1], Chen Dong[2], Dai Hongbing[1], Wang Xinhua[2,3]

(1. School of Information Science and Engineering, Yunnan University, Kunming 650504. China;

2. Yunnan Astronomical Observatory, Chinese Academy of Science, Kunming 650011 China;

3. University of Chinese Academy of Science, Beijing 100049 China)

Abstract: At present, speckle interferometry and speckle masking are often used to reconstruct the mode and phase of the target in high-resolution solar image reconstruction. However, due to a large amount of grouped and partitioned data and the complexity of the algorithm, it is difficult to meet the requirements of real-time reconstruction. In order to alleviate the pressure of data processing, based on the existing CPU/GPU hybrid computing method of single block data, a method of allocating multi-component block data to GPU through multi-process and parallel processing is proposed. The experiment shows that the method based on a multi-process parallel acceleration method can improve the utilization efficiency of CPU and GPU resources, enable GPU to process multi-block data at the same time, and significantly improve the speed of image block processing with an acceleration ratio of about 4.7. Relevant studies can provide a reference for the parallel processing of astronomical data.

Key words: Multi-process; Parallel Computing; Image Reconstruction; Speckle Masking; Speckle Interferometry